A computational study of a solver system for processing two-stage stochastic linear programming problems *

Victor Zverovich Csaba Fábián Francis Ellison
Gautam Mitra

July 2, 2009

1 Introduction and background

Formulation of stochastic optimisation problems and computational algorithms for their solution continue to make steady progress as can be seen from an analysis of many developments in this field. The edited volume by Wallace and Ziemba (2005) outlines both the SP modelling systems and many applications in diverse domains.

More recently, Fabozzi et al. (2007) has considered the application of SP models to challenging financial engineering problems. The tightly knit yet highly focused group of researchers COSP: Committee on Stochastic Programming, their triennial international SP conference, and their active website points to the progressive acceptance of SP as a valuable decision tool. At the same time many of the major software vendors, namely, XPRESS, AIMMS, MAXIMAL, and GAMS have started offering SP extensions to their optimisation suites.

Our analysis of the modelling and algorithmic solver requirements reveals that (a) modelling support (b) scenario generation and (c) solution methods are three important aspects of a working SP system. Our research is focussed on all three aspects and we refer the readers to Valente et al. (2009) for modelling and Mitra et al. (2007) and Di Domenica et al. (2009) for scenario generation. In this paper we are concerned entirely with computational solution methods. Given the tremendous advance in LP solver

*CARISMA, Brunel University, Uxbridge, Middlesex, United Kingdom

algorithms there is certain amount of complacency that by constructing a "deterministic equivalent" problems it is possible to process most realistic instances of SP problems. In this paper we highlight the shortcoming of this line of argument. We describe the implementation and refinement of established algorithmic methods and report a computational study which clearly underpins the superior scale up properties of the solution methods which are described in this paper.

A taxonomy of the important class of SP problems may be found in Valente et al. (2008, 2009). The most important class of problems with many applications is the two-stage stochastic programming model with recourse, which originated from the early research of Dantzig (1955), Beale (1955) and Wets (1974).

A comprehensive treatment of the model and solution methods can be found in Kall and Wallace (1994), Prékopa (1995), Birge and Louveaux (1997), Mayer (1998), Ruszczyński and Shapiro (2003), and Kall and Mayer (2005). Some of these monographs contain generalisations of the original model. Colombo et al. (2006) and Gassmann and Wallace (1996) describe computational studies which are based on interior point method and simplex based methods respectively.

The rest of this paper is organised in the following way. In section 2 we introduce the model setting of the two stage stochastic programming problem, in section 3 we consider a selection of solution methods for processing this class of problems. The established approaches of processing the deterministic equivalent LP form, the decomposition approach of Benders, the need for regularisation are also discussed. We also introduce the concept of level decomposition and explain how it fits into the concept of regularisation. In section 4 we set out the computational study and in section 5 we summarise our conclusions.

2 The model setting

2.1 Two-stage problems

In this paper we assume that the random parameters have a discrete finite distribution, and we only consider linear models. This class is based on two key concepts of (i) a finite set of discrete scenarios (of model parameters) and (ii) a partition of variables to first stage ("here and now") decisions variables and a second stage observation of the parameter realisations and corrective actions and the corresponding recourse (decision) variables.

The result of the first decision will be represented by the vector $\boldsymbol{x}$. As-

sume there are S possible outcomes of the random event (*scenarios*), the ith outcome occurring with probability p_i. Suppose the first decision has been made with the result $\boldsymbol{x}$, and the ith scenario has realised. The second decision will be represented by the following *second-stage problem* or *recourse problem* that we denote by $R_i(\boldsymbol{x})$:

$$\begin{array}{rl} \min & \boldsymbol{q_i^T y} \\ \text{subject to} & T_i\boldsymbol{x} + W_i\boldsymbol{y} = \boldsymbol{h_i}, \\ & \boldsymbol{y} \geq \boldsymbol{0}, \end{array} \tag{1}$$

where $\boldsymbol{q_i}$, $\boldsymbol{h_i}$ are given vectors and T_i, W_i are given matrices of compatible sizes. The decision variable is $\boldsymbol{y}$. Let K_i denote the set of those $\boldsymbol{x}$ vectors for which the recourse problem $R_i(\boldsymbol{x})$ has a feasible solution. This is a convex polyhedron. For $\boldsymbol{x} \in K_i$, let $q_i(\boldsymbol{x})$ denote the optimal objective value of the recourse problem. We assume that $q_i(\boldsymbol{x}) > -\infty$. (Or equivalently, we assume that the dual of the recourse problem $R_i(\boldsymbol{x})$ has a feasible solution. Solvability of the dual problem does not depend on $\boldsymbol{x}$.) The function $q_i : \ K_i \to \mathbb{R}$ is a *polyhedral* (i.e., piecewise linear) convex function.

The customary formulation of the *first-stage problem* that represents the first decision, is as follows:

$$\begin{array}{rl} \min & \boldsymbol{c^T x} + \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x}) \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}, \\ & \boldsymbol{x} \in K_i \ \ (i = 1, \ldots, S), \end{array} \tag{2}$$

where $\boldsymbol{c}$ and $\boldsymbol{b}$ are given vectors and A is a given matrix, with compatible sizes. We assume that $X := \{\boldsymbol{x} \,|A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\}$ is a non-empty bounded polyhedron. The expectation in the objective, $Q(\boldsymbol{x}) := \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x})$, is called the *expected recourse function*. This is a polyhedral convex function with the domain $K := K_1 \cap \ldots \cap K_S$.

The two-stage stochastic programming problem (2)-(1) can be formulated as a single linear programming problem called the *deterministic equivalent problem*:

$$\begin{array}{rccccccccc} \min & \boldsymbol{c^T x} & + & p_1\boldsymbol{q_1^T y_1} & + & \ldots & + & p_S\boldsymbol{q_S^T y_S} & & \\ \text{subject to} & A\boldsymbol{x} & & & & & & & = & \boldsymbol{b}, \\ & T_1\boldsymbol{x} & + & W_1\boldsymbol{y_1} & & & & & = & \boldsymbol{h_1}, \\ & \vdots & & & & \ddots & & & & \vdots \\ & T_S\boldsymbol{x} & + & & & & & W_S\boldsymbol{y_S} & = & \boldsymbol{h_S}, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{y_1} \geq \boldsymbol{0}, \ \ldots, \ \boldsymbol{y_S} \geq \boldsymbol{0}. & & & & & & & & & \end{array} \tag{3}$$

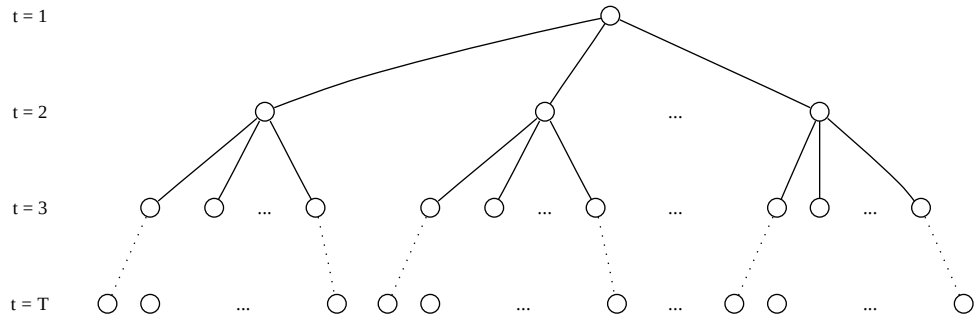

Figure 1: Scenario tree of a multistage problem

2.2 Two-stage approximations of multistage problems

The solution methods for multistage stochastic programming problems were out of scope of the current paper. However we decided to consider two-stage approximations of multistage problems. In order to describe the approximation method let us first consider the following formulation of a multistage SLP problem:

$$\begin{array}{rl} \min & \sum_{i=1}^{S} p_i[\boldsymbol{c_1^T x_1^i} + \boldsymbol{(c_2^i)^T x_2^i} + \boldsymbol{(c_3^i)^T x_3^i} + \cdots + \boldsymbol{(c_T^i)^T x_T^i}] \\ \text{subject to} & A_{11}\boldsymbol{x_1^i} = \boldsymbol{b_1}, \\ & A_{21}^i\boldsymbol{x_1^i} + A_{22}^i\boldsymbol{x_2^i} = \boldsymbol{b_2^i}, \\ & \cdots \\ & A_{T1}^i\boldsymbol{x_1^i} + A_{T2}^i\boldsymbol{x_2^i} + \cdots + A_{TT}^i\boldsymbol{x_T^i} = \boldsymbol{b_T^i}, \\ & \boldsymbol{x_t^i} \geq \boldsymbol{0} \ \ (t = 1, \ldots, T), \\ & \boldsymbol{x_t^i} \in \mathbb{R}^{n_t} \ \ (t = 1, \ldots, T), \\ & \boldsymbol{x_t^j} = \boldsymbol{x_t^k} \text{ for all } j, k \text{ such that } (\boldsymbol{\xi_1^j}, \ldots, \boldsymbol{\xi_t^j}) = (\boldsymbol{\xi_1^k}, \ldots, \boldsymbol{\xi_t^k}), \end{array} \tag{4}$$

where $\boldsymbol{c_1}, \boldsymbol{b_1}, \boldsymbol{c_t^i}, \boldsymbol{b_t^i}$ $(t = 2, \ldots, T)$ are given vectors, $A_1, A_{t1}^i, A_{t2}^i, \ldots, A_{tt}^i$ $(t = 2, \ldots, T)$ are given matrices of compatible sizes, $\boldsymbol{\xi_1^i} = (c_1, b_1, A_{11})$ and $\boldsymbol{\xi_t^i} = (c_t^i, b_t^i, A_{t1}^i, A_{t2}^i, \ldots, A_{tt}^i)$ $(t = 2, \ldots, T)$.

Scenario tree of a multistage problem is shown in Figure 1. It graphically represents nonanicipativity restrictions imposed on decision variables.

We approximated multistage problems with two-stage problems by relaxing nonanticipativity constraints in (4) for stages $2, ..., T$ and aggregating

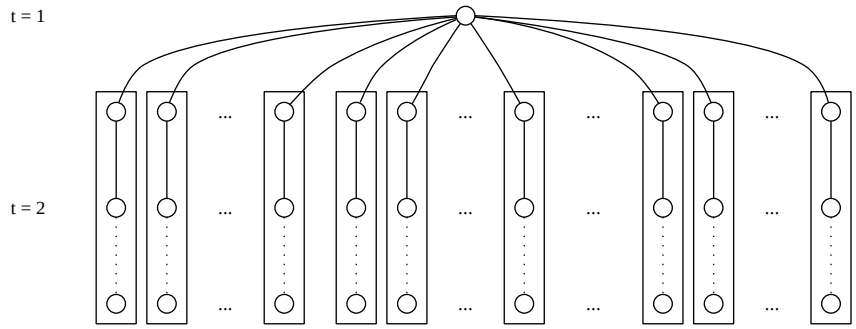

Figure 2: Scenario tree of a two-stage approximation problem

these stages into one which resulted in the following two-stage formulation:

$$\begin{array}{rl} \min & \sum_{i=1}^{S} p_i[\boldsymbol{c_1^T x_1^i} + \boldsymbol{(q^i)^T y^i}] \\ \text{subject to} & A_{11}\boldsymbol{x_1^i} \qquad\quad = \boldsymbol{b_1}, \\ & T^i\boldsymbol{x_1^i} + W^i\boldsymbol{y^i} = \boldsymbol{h^i}, \\ & \boldsymbol{x_1^i} \geq \boldsymbol{0}, \boldsymbol{x_1^i} \in \mathbb{R}^{n_1}, \\ & \boldsymbol{y^i} \geq \boldsymbol{0}, \boldsymbol{y^i} \in \mathbb{R}^{n_2+\cdots+n_T}, \end{array} \tag{5}$$

where

$$T^i = \begin{pmatrix} A_{21}^i \\ A_{31}^i \\ \vdots \\ A_{T1}^i \end{pmatrix}, W^i = \begin{pmatrix} A_{22}^i & & & \\ A_{32}^i & A_{33}^i & & \\ \vdots & \vdots & \ddots & \\ A_{T2}^i & A_{T3}^i & \ldots & A_{TT}^i \end{pmatrix},$$

$$\boldsymbol{y^i} = (\boldsymbol{x_2^i}, \ldots, \boldsymbol{x_T^i}), \boldsymbol{h^i} = (\boldsymbol{b_2^i}, \ldots, \boldsymbol{b_T^i}).$$

The scenario tree of the approximation problem is shown on Figure 2 where the rectangular regions represent nodes formed by aggregating nodes of the original tree along the paths from the children of the root to the leaves.

3 A selection of methods

3.1 Solution of the deterministic equivalent by simplex and interior-point methods

The first approach to solve stochastic linear programming problems we considered was using a state-of-the-art LP solver to optimize deterministic equiv-

alent problem (3). For this purpose CPLEX barrier and dual simplex optimizers were selected since they provide high-performance implementation of corresponding methods.

3.2 Decomposition

The deterministic equivalent problem (3) is a linear programming problem of a specific structure: For each scenario, a subproblem is included that describes the second-stage decision in case this scenario realises. The subproblems are linked by the first-stage decision variables. Dantzig and Madansky (1961) observed that the dual of the deterministic equivalent problem fits the prototype for the Dantzig-Wolfe decomposition (1960).

Van Slyke and Wets (1969) proposed a cutting-plane approach for the first-stage problem (2). Their L-Shaped method builds respective cutting-plane models of the feasible domain $K = K_1 \cap \ldots \cap K_S$ and of the expected recourse function $Q = \sum_{i=1}^{S} p_i q_i$. We outline cutting-plane models and their relationship with decomposition.

Let us denote the dual of $R_i(\boldsymbol{x})$ in (1) by $D_i(\boldsymbol{x})$:

$$\begin{array}{rl} \max & \boldsymbol{z^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{z^T} W_i \leq \boldsymbol{q_i^T}. \end{array} \tag{6}$$

The feasible region is a convex polyhedron that we assumed nonempty. We will characterise this polyhedron by two finite sets of vectors: let U_i and V_i denote the sets of the extremal points and of the extremal rays, respectively, in case the polyhedron can be represented by these sets. – To handle the general case, some more formality is needed: Let us add a slack vector $\boldsymbol{s}$ of appropriate dimension, and use the notation $[W_i^T, I](\boldsymbol{z}, \boldsymbol{s}) = W_i^T\boldsymbol{z} + \boldsymbol{s}$. Given a composite vector $(\boldsymbol{z}, \boldsymbol{s})$ of appropriate dimensions, let $\text{support}(\boldsymbol{z}, \boldsymbol{s})$ denote the set of those column-vectors of the composite matrix $[W_i^T, I]$ that belong to non-zero $(\boldsymbol{z}, \boldsymbol{s})$-components. Using these, let

$$\begin{aligned} U_i := & \left\{ \boldsymbol{z} \,\middle|\, W_i^T\boldsymbol{z} + \boldsymbol{s} = \boldsymbol{q_i}, \boldsymbol{s} \geq \boldsymbol{0}, \text{ support}(\boldsymbol{z}, \boldsymbol{s}) \text{ is a linearly independent set} \right\}, \\ V_i := & \left\{ \boldsymbol{z} \,\middle|\, W_i^T\boldsymbol{z} + \boldsymbol{s} = \boldsymbol{0}, \boldsymbol{s} \geq \boldsymbol{0}, \text{ support}(\boldsymbol{z}, \boldsymbol{s}) \text{ is a minimal dependent set} \right\}. \end{aligned}$$

These are finite sets, and the feasible domain of the dual problem $D_i(\boldsymbol{x})$ in (6) can be represented as convex combinations of U_i-elements added to cone-combinations of V_i-elements.

We have $\boldsymbol{x} \in K_i$ if and only if the dual problem $D_i(\boldsymbol{x})$ has finite optimum, i.e.,

$$\boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq 0 \quad \text{holds for every } \boldsymbol{v_i} \in V_i.$$

In this case, the optimum of $D_i(\boldsymbol{x})$ is attained at an extremal point, and can be computed as

$$\begin{array}{rl} \min & \vartheta_i \\ \text{subject to} & \vartheta_i \in \mathbb{R}, \\ & \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq \vartheta_i \quad (\boldsymbol{u_i} \in U_i). \end{array}$$

According to the linear programming duality theorem, the optimum of the above problem is equal to $q_i(\boldsymbol{x})$. Hence the first-stage problem (2) can be written as

$$\begin{array}{rll} \min & \boldsymbol{c^T x} + \sum_{i=1}^{S} p_i \vartheta_i & \\ \text{subject to} & \boldsymbol{x} \in X, \quad \vartheta_i \in \mathbb{R} & (i = 1, \ldots, S), \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq 0 & (\boldsymbol{v_i} \in V_i,\ i = 1, \ldots, S), \\ & \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq \vartheta_i & (\boldsymbol{u_i} \in U_i,\ i = 1, \ldots, S). \end{array} \tag{7}$$

The *aggregate form* of the above problem is

$$\begin{array}{rll} \min & \boldsymbol{c^T x} + \vartheta & \\ \text{subject to} & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, & \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq 0 & (\boldsymbol{v_i} \in V_i,\ i = 1, \ldots, S), \\ & \sum_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq \vartheta & \big((\boldsymbol{u_1}, \ldots, \boldsymbol{u_S}) \in \mathcal{U}\big), \end{array} \tag{8}$$

where $\mathcal{U} \subset U_1 \times \cdots \times U_S$ is such a subset that contains an element for each facet in the graph of the polyhedral convex function Q; formally, we have

$$Q(\boldsymbol{x}) \;=\; \sum_{i=1}^{S} \left\{ p_i \max_{\boldsymbol{u_i} \in U_i} \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \right\} \;=\; \max_{(\boldsymbol{u_1}, \ldots, \boldsymbol{u_S}) \in \mathcal{U}} \sum_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}).$$

Cutting-plane methods can be devised on the basis of either the *disaggregate* formulation (7) or the aggregate formulation (8). These are iterative methods that build respective cutting-plane models of the feasible set K and of the expected recourse function Q. The cuts belonging to the model of K are called *feasibility cuts*, and those belonging to the model of Q are called *optimality cuts*. Cuts at a given iterate $\widehat{\boldsymbol{x}}$ can be generated by solving the dual problems $D_i(\widehat{\boldsymbol{x}})$ $(i = 1, \ldots, S)$. Problems with unbounded objectives yield feasibility cuts, and problems with optimal solutions yield optimality cuts.

In its original form, the L-Shaped method of Van Slyke and Wets (1969) works on the aggregate problem. A *multicut* version that works on the disaggregate problem was proposed by Birge and Louveaux (1988).

There is a close relationship between decomposition and cutting-plane approaches. Actually it turns out that the following approaches yield methods that are in principle identical:

– cutting-plane method for either the disaggregate problem (7) or the aggregate problem (8),

– Dantzig-Wolfe decomposition (1960) applied to the dual of the deterministic equivalent problem (3),

– Benders decomposition (1962) applied to the deterministic equivalent problem (3).

Technical differences between different approaches may result in substantial differences in efficiency. Cutting-plane formulations have the advantage that they give a clear visual impression of the procedure. A state-of-the-art overview of decomposition methods can be found in Ruszczyński (2003).

Remark 1 *Since classic methods handle feasibility issues by imposing feasibility cuts, the scope of optimisation may alternate between minimising the objective function and finding a solution that satisfies existing cuts.*

Implementation: aggregate model

For the present computational study, we implemented a decomposition method that works on the aggregate problem. After a certain number of iterations, let $\widehat{V}_i \subset V_i$ denote the subsets of the known elements of V_i ($i = 1, \ldots, S$), respectively. Similarly, let $\widehat{\mathcal{U}} \subset \mathcal{U}$ denote the subset of the known elements of $\mathcal{U} \subset U_1 \times \cdots \times U_S$. We solve the current model problem

$$\begin{array}{rll} \min & \boldsymbol{c^T x} + \vartheta & \\ \text{subject to} & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, & \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq 0 & (\boldsymbol{v_i} \in \widehat{V}_i,\ i = 1, \ldots, S), \\ & \sum_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \leq \vartheta & \big((\boldsymbol{u_1}, \ldots, \boldsymbol{u_S}) \in \widehat{\mathcal{U}}\big). \end{array} \tag{9}$$

If the model problem is infeasible, then so is the original problem. Let $\widehat{\boldsymbol{x}}$ denote an optimal solution. In order to generate cuts at $\widehat{\boldsymbol{x}}$, we solve the dual recourse problems $D_i(\widehat{\boldsymbol{x}})$ ($i = 1, \ldots, S$) with a simplex-type method. Let

$$\widehat{I} := \{1 \leq i \leq S \,|\, \text{problem } D_i(\widehat{\boldsymbol{x}}) \text{ has unbounded objective}\}.$$

If $\widehat{I} = \emptyset$ then the solution process of each dual recourse problem terminated with an optimal basic solution $\widehat{\boldsymbol{u}}_{\boldsymbol{i}} \in U_i$. If $\widehat{\boldsymbol{x}}$ is near-optimal then the procedure stops. Otherwise we add the point $(\widehat{\boldsymbol{u}}_{\mathbf{1}}, \ldots, \widehat{\boldsymbol{u}}_{\boldsymbol{S}})$ to $\widehat{\mathcal{U}}$, rebuild the model problem (9), and start a new iteration.

If $\widehat{I} \neq \emptyset$ then for $i \in \widehat{I}$, the solution process of the dual recourse problem $D_i(\widehat{\boldsymbol{x}})$ terminated with $\widehat{\boldsymbol{v}}_{\boldsymbol{i}} \in V_i$. We add $\widehat{\boldsymbol{v}}_{\boldsymbol{i}}$ to $\widehat{V}_i$ $(i \in \widehat{I})$, rebuild the model problem (9), and start a new iteration.

3.3 Heuristic regularization: straight dampening of jumps in pure Benders decomposition using adaptive weights

It is observed that successive iterations do not generally produce an orderly progression of solutions - in the sense that while the change in objective value from one iteration to the next may be very small - zero even - a wide difference can occur between corresponding values of the first-stage variables - x. It has generally been the aim of writers such as Ruszczyński (1986), Lemaréchal et al. (1995), Fábián (2000) to find a method of 'Regularising' - that is smoothing out these changes so that the optimum can be reached in fewer iterations without extensive jumping around the near-optimal region of each first-stage problem.

In discussions between H.Gassman and F.Ellison, Gassman observed that the easiest and most obvious answer was to select a point between the previous iterate and the current solution by linear interpolation. Thus, given iterate $\overline{x}_{k-1}$ and solution x_k to iterations $k-1$ and k of the algorithm, the next iterate to choose with which to start iteration $k+1$ would be:

$$\overline{x}_k := \alpha.x_k + (1-\alpha)\overline{x}_{k-1}$$

where α is a value between zero and one. Values less than some minimum α^{min} are deprecated ($\alpha^{min} = 0.5$ is likely) and we seek a formula:

$$\alpha = 1.0 + (\alpha^{min} - 1.0)(\frac{D}{D^{max}})^p$$

where D is some factor expressing the difference between $\overline{x}_{k-1}$ and x_k, D^{max} is the maximum of D, and p is some power not less than one.

The obvious function to use for D would be the distance between the two solution vectors - that is the 2-norm $||x_k - \overline{x}_{k-1}||_2$. However in this case a problem arises in determining D^{max}, and we look for another answer.

Solution algorithms based on the Simplex method (this includes IPM followed by Crossover) determine a 'basis' - that is a vector of codes describing

the status of each constraint and each variable in the solution. Consider the vector $b_k = \{b_{j,k}\}$ of basis codes for the variables $x_j, j = 1, ..., n$ in the first-stage solution of iteration k. Determine a difference-vector $\Delta_k = \{\delta_{j,k}\}$, where $\delta_{j,k} = 0$ if $b_{j,k} = b_{j,k-1}$ and $\delta_{j,k} = 1$ otherwise. Then the function D is just $\sum(\delta_{j,k})$, its maximum D^{max} cannot exceed n, and the formula for α is established on assuming some value for the power p ($p = 1$ is likely).

3.4 Regularisation

We have referred to the habit of zigzagging in cutting plane methods, which is the consequence of using linear approximation. Improved methods were developed that use quadratic approximation: proximal point method by Rockafellar (1976), and bundle method by Lemaréchal (1978) and Kiwiel (1985).

The Regularized Decomposition (RD) method of Ruszczyński (1986) is a bundle-type method applied to the minimisation of the sum of polyhedral convex functions over a convex polyhedron, hence this method fits the disaggregate problem (7). The RD method lays an emphasis on keeping the master problem small. (This is achieved by an effective constraint reduction strategy.) A recent discussion of the RD method can be found in Ruszczyński (2003).

The RD method, like the unregularised methods discussed in Section 3.2, handles feasibility issues by imposing feasibility cuts. Hence the scope alternation mentioned in Remark 1 may occur with this method also.

Ruszczyński and Świętanowski (1997) implemented the RD method, and solved two-stage stochastic programming problems, each with a growing scenario set. Their test results show that the number of the RD master iterations required is a slowly increasing function of the number of the scenarios.

A more recent development in convex programming methods is the level methods of Lemaréchal, Nemirovskii, and Nesterov (1995). These are special bundle-type methods that use level sets of the model functions for regularisation.

The unconstrained level method minimises a convex function over a convex bounded polyhedron. Lemaréchal, Nemirovskii, and Nesterov prove the following efficiency estimate: to obtain an ϵ-optimal solution, the method needs no more than

$$\kappa \left(\frac{DL}{\epsilon}\right)^2 \tag{10}$$

iterations, where D is the diameter of the feasible polyhedron, L is a Lipschitz constant of the objective function, and κ is a constant that depends only on the parameter of the algorithm.

The constrained level method minimises a convex objective function over the intersection of a level set of a convex constraint function and a convex bounded polyhedron. Lemaréchal, Nemirovskii, and Nesterov prove the following efficiency estimate: to obtain an ϵ-optimal solution, the method needs no more than

$$\kappa \left(\frac{DL}{\epsilon}\right)^2 \log\left(\frac{DL}{\epsilon}\right) \tag{11}$$

iterations, where D is the diameter of the feasible polyhedron, L is a common Lipschitz constant of the objective and constraint functions, and κ is a constant that depends only on the parameters of the method.

Fábián (2000) developed inexact versions of the level methods. These methods use approximate data to construct models of the objective and constraint functions. At the beginning of the procedure, a rough approximation is used. As the optimum is approached, the accuracy is gradually increased. Fábián proved that the efficiency estimates (10)-(11) also hold for the inexact methods.

Fábián and Szőke (2007) adapted the inexact level methods to the solution of two-stage stochastic programming problems. The resulting Level Decomposition (LD) method handles feasibility and optimality issues simultaneously, in a unified manner. Second-stage infeasibility is controlled through a constraint function rather than infeasibility cuts. Hence regularisation extends to feasibility issues, and there is no scope alternation as mentioned in Remark 1. Moreover the LD framework allows a progressive approximation of the distribution, and approximate solution of the second-stage problems. (In contrast to classic frameworks, LD does not require optimal basic solutions for the second-stage problems, as convergence is ensured by level method mechanism.)

Fábián and Szőke solved two-stage stochastic programming problems with growing scenario sets. Their results show that the number of LD master iterations is independent of the number of the scenarios. Moreover the methods performed much better then the theoretical efficiency estimates (10)-(11) might suggest. Problems having complete recourse were solved using the unconstrained method. (There being no second-stage infeasibility, there was no need for a constraint function to control it.) Solving a complete-recourse problem with different settings of the stopping tolerance ϵ, the number of the LD master iterations was found to be proportional with $\log(1/\epsilon)$.

Implementation: regularisation of the expected recourse function by level method

For the present computational study, we implemented a rudimentary version of the level decomposition. We use the original exact level method, hence we use no distribution approximation, and second-stage problems are solved exactly (i.e., with the same high accuracy always). So far we have implemented the unconstrained method and control second-stage infeasibility by imposing feasibility cuts, the way classic methods do. Hence regularisation does not presently extend to feasibility issues and the scope alternation mentioned in Remark 1 does occur.

Our computational results reported in section 4.3 show that level-type regularisation is indeed advantageous.

4 Computational study

4.1 Experimental setup

The computational experiments were performed on a Windows XP machine with Intel CORE2 2.4 GHz CPU and 3 GB of RAM. Deterministic equivalents were solved with CPLEX 11.0 dual simplex and barrier optimizers. Crossover to a basic solution was disabled for the barrier optimizer, for other CPLEX options the default values were used.

The times are reported in seconds with times of reading input files included. For simplex and IPM the times of constructing deterministic equivalent problems are also included though it should be noted that they only amount to small fractions of the total. FortMP linear and quadratic programming solver described by Ellison et al. (2008) was used to solve master problem and subproblems in the implementations of Benders decomposition and level method. All the test problems were presented in SMPS format introduced by Birge et al. (1987).

The first-stage solution of the expected value problem was taken as a starting point for the decomposition methods. The value of the parameter λ for level decomposition was set to 0.5. The values of the parameters α^{min} and p for the heuristic regularisation method were set to 0.5 and 1 respectively.

The deterministic equivalents were constructed and passed to CPLEX through Open Solver Interface (OSI). This interface together with implementations for various solvers is provided by an open-source project of the same name which is a part of COIN-OR repository (Lougee-Heimer, 2003). It allows uniform interaction with supported solvers and in particular with CPLEX and FortMP solvers which we used in our study.

4.2 Data sets

We considered test problems which were drawn from four different sources described in Table 1. Tables 2 – 6 give the parameters of these problems. Columns A and W of these tables give the dimensions of corresponding matrices in the following formulation of a two-stage SP problem:

$$\begin{array}{rl} \min & \boldsymbol{c^T x} + \mathbb{E}Q(\boldsymbol{x}, \boldsymbol{\xi}) \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{array}$$

where

$$\begin{array}{rl} Q(\boldsymbol{x}, \boldsymbol{\xi}) = \min & \boldsymbol{q^T y} \\ \text{subject to} & W\boldsymbol{y} = \boldsymbol{h} - T\boldsymbol{x}, \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{array}$$

and vector $\boldsymbol{\xi}$ is composed of the random components of $\boldsymbol{h}, T, W$ and $\boldsymbol{q}$.

NR and NNZ denote the number of random elements and the number of nonzero matrix elements respectively. Optimal values reported in column Opt were obtained using level method. For the WATSON problems the optimal values of their two-stage approximations are specified in this column.

Most problems considered in this paper have stochasticity only in the right-hand side (RHS). The exceptions are SAPHIR and WATSON families of problems which have random elements both in the RHS and the matrix.

It should be noted that the problems generated with GENSLP do not possess any internal structure inherent in real-world problems. However they are still useful for the purposes of comparing scale-up properties of algorithms.

4.3 Computational results

The computational results are presented in Tables 7 – 11. Iter denotes the number of iterations. For Benders decomposition, level method and linear damping these are the numbers of master iterations.

Finally we present the results in the form of performance profiles. The performance profile for a solver is defined by Dolan and Moré (2002) as the cumulative distribution function for a performance metric. We use the ratio of the solving time versus the best time as the performance metric. Let P and M be the set of problems and the set of solution methods respectively. We define by $t_{p,m}$ the time of solving problem $p \in P$ with method $m \in M$. For every pair (p, m) we compute performance ratio

$$r_{p,m} = \frac{t_{p,m}}{\min\{t_{p,m} | m \in M\}},$$

Source	Reference	Comments
1. POSTS collection	Holmes (1995)	Two-stage problems from the (PO)rtable (S)tochastic programming (T)est (S)et (POSTS)
2. Slptestset collection	Ariyawansa and Felt (2004)	Two-stage problems from the collection of stochastic LP test problems
3. Random problems	Kall and Mayer (1998)	Artificial test problems generated with pseudo random stochastic LP problem generator GENSLP
4. SAMPL problems	König et al. (2007), Valente et al. (2008)	Problems instantiated from the SAPHIR gas portfolio planning model formulated in Stochastic AMPL (SAMPL)
5. WATSON problems	Consigli and Dempster (1998)	WATSON pension fund management test problems

Table 1: Sources of test problems

					Deterministic Equivalent		
Name	A	W	NR	Scen	Matrix	NNZ	Opt
pltexpA2	62×188	104×272	7	6	686×1820	3703	-9.47935
				16	1726×4540	9233	-9.66234
fxm2	92×114	238×343	2	6	1520×2172	12139	18416.8
				16	3900×5602	31239	18416.8
stormG2	185×121	528×1259	117	8	4409×10193	27424	15535236
				27	14441×34114	90903	15508982
				125	66185×157496	418321	15512091
				1000	528185×1259121	3341696	15802590

Table 2: Parameters of test problems from POSTS collection

Name	A	W	NR	Scen	Deterministic Equivalent Matrix	NNZ	Opt
AIRL2	2×4	6×8	2	25	152×204	604	269665
LandS	2×4	7×12	1	3	23×40	92	381.853
4node	14×52	74×186	3	2^4	1198×3028	7743	423.012
			4	2^5	2382×6004	15231	423.013
			5	2^6	4750×11956	30207	423.012
			5	2^7	9486×23860	60159	423.012
			6	2^8	18958×47668	120063	425.375
			7	2^9	37902×95284	239871	429.962
			8	2^{10}	75790×190516	479487	434.113
			9	2^{11}	151566×380980	958719	441.738
			12	2^{12}	303118×761908	1917183	446.856
			12	2^{13}	606222×1523764	3834111	446.856
			12	2^{14}	1212430×3047476	7667967	446.856
			12	2^{15}	2424846×6094900	15335679	446.856

Table 3: Parameters of test problems from Slptestset collection

Name	A	W	NR	Scen	Deterministic Equivalent Matrix	NNZ	Opt
rand0	50×100	25×50	25	2×10^3	50050×100100	754501	162.146
				4×10^3	100050×200100	1508501	199.032
				6×10^3	150050×300100	2262501	140.274
				8×10^3	200050×400100	3016501	170.318
				10^4	250050×500100	3770501	139.129
rand1	100×200	50×100	50	2×10^3	100100×200200	3006001	244.159
				4×10^3	200100×400200	6010001	259.346
				6×10^3	300100×600200	9014001	297.562
				8×10^3	400100×800200	12018001	262.451
				10^4	500100×1000200	15022001	298.638
rand2	150×300	75×150	75	5×10^3	150150×300300	6758501	209.151
				4×10^3	300150×600300	13512501	218.247
				6×10^3	450150×900300	20266501	239.720
				8×10^3	600150×1200300	27020501	239.158
				10^4	750150×1500300	33774501	231.706

Table 4: Parameters of test problems generated with GENSLP

Name	A	W	NR	Scen	Deterministic Equivalent Matrix	NNZ	Opt
saphir	32×53	8678×3924	958	50	433932×196253	1136753	129506233
				100	867832×392453	2273403	129059362
				200	1735632×784853	4546703	141473266
				500	4339032×1962053	11366603	137871740
				1000	8678032×3924053	22733103	133036857

Table 5: Parameters of SAMPL problems

Name	A	W	NR	Scen	Deterministic Equivalent Matrix	NNZ	Opt
WATSON.I	11×15	324×587	536	128	41483×75151	188828	-2271.17866
				256	82955×150287	377628	-2733.63695
				512	165899×300559	755228	-2810.75153
				1024	331787×601103	1510428	-2750.48955

Table 6: Parameters of two-stage approximations of WATSON problems

If method m failed to solve problem p the formula above is not defined. In this case we set $r_{p,m} := \infty$.

The cumulative distribution function for the performance ratio is defined as follows:

$$\rho_m(\tau) = \frac{|\{p \in P | r_{p,m} \leq \tau\}|}{|P|}$$

We calculated performance profile of each considered method on the whole set of test problems. These profiles are shown in Figure 3. The value of $\rho_m(\tau)$ gives the probability that method m solves a problem within a ratio τ of the best solver. For example according to Figure 3 level method was the first in more than 30% of cases and solved all the problems within a ratio 6 of the best time.

The notable advantages of performance profiles over other approaches to performance comparison are as follows. Firstly, they minimize the influence of a small subset of problems on the benchmarking process. Secondly, there is no need to discard solver failures. Thirdly, performance profiles provide a visualisation of large sets of test results as we have in our case.

As can be seen from Figure 3, while in most cases the performance of CPLEX barrier optimizer is better it was not able to solve some of the problems. Several large instances were not solved due to high memory requirements of constructing and solving deterministic equivalent. Other failures were caused by numerical difficulties. The performance profiles of pure Benders decomposition and linear damping are very similar to each other and the level method profile dominates both of them.

4.4 Comments on scale-up properties and on accuracy

We performed a set of experiments with different stopping tolerances up to the tolerance 10^{-7}. Figures 4 – 7 show typical patterns of the change in the gap between lower and upper bounds on objective function.

The computational results given in the previous section where obtained using the relative stopping tolerance $\varepsilon = 10^{-5}$ for decomposition methods,

Scen	IPM Time	IPM Iter	Simplex Time	Simplex Iter	Benders Time	Benders Iter	Level Time	Level Iter	Lin. Damp. Time	Lin. Damp. Iter
pltexpA2										
6	0.06	14	0.15	329	0.04	1	0.03	1	0.03	1
16	0.13	16	0.17	810	0.08	4	0.10	4	‡	‡
fxm2										
6	0.09	17	0.24	1281	0.29	23	0.35	15	0.36	25
16	0.20	23	0.47	3374	0.39	22	0.53	18	0.43	23
stormG2										
8	0.38	28	0.32	3675	0.60	23	0.83	22	0.67	25
27	3.33	27	0.87	13128	1.93	30	1.65	22	2.13	32
125	12.33	57	7.00	71611	8.38	32	4.99	19	9.75	37
1000	189.53	109	305.81	758078	80.20	41	34.46	18	81.07	41

Table 7: Solution times and iteration counts for POSTS problems
‡ Failed to solve due to numerical difficulties

Scen	IPM Time	IPM Iter	Simplex Time	Simplex Iter	Benders Time	Benders Iter	Level Time	Level Iter	Lin. Damp. Time	Lin. Damp. Iter
AIRL2										
25	0.04	11	0.14	145	0.08	10	0.16	16	0.09	12
LandS										
3	0.04	9	0.11	21	0.01	8	0.04	9	0.01	8
4node										
2^4	0.19	17	0.20	1461	1.44	102	1.18	47	‡	‡
2^5	0.65	15	0.37	3244	3.60	130	1.87	66	‡	‡
2^6	0.70	17	0.88	6847	6.79	135	2.36	54	3.74	85
2^7	0.71	26	2.48	13498	10.25	115	3.37	45	6.56	79
2^8	1.53	30	9.88	27743	16.17	101	8.75	60	‡	‡
2^9	3.38	30	41.74	54861	34.04	109	18.08	67	37.32	115
2^{10}	7.51	32	457.53	130701	69.13	110	36.34	68	52.60	86
2^{11}	17.93	36	1262.75	239159	240.25	184	63.28	59	208.20	162
2^{12}	44.95	45	11733.86	475971	538.26	215	129.57	63	502.53	193
2^{13}	79.73	45	*	*	1474.48	286	229.72	56	‡	‡
2^{14}	†	†	†	†	1850.52	194	459.27	58	1240.92	132
2^{15}	†	†	†	†	5785.07	279	1029.74	65	4320.99	211

Table 8: Solution times and iteration counts for Slptestset problems
* Failed to solve due to timeout
† Failed to solve due to insufficient memory
‡ Failed to solve due to numerical difficulties

Scen	IPM Time	IPM Iter	Simplex Time	Simplex Iter	Benders Time	Benders Iter	Level Time	Level Iter	Lin. Damp. Time	Lin. Damp. Iter
rand0										
2×10^3	16.71	44	541.78	84571	62.68	80	33.73	42	59.33	77
4×10^3	30.11	40	2632.72	155926	112.91	72	59.41	37	89.01	58
6×10^3	56.90	52	8688.20	257614	287.09	124	137.20	58	236.76	103
8×10^3	83.35	57	*	*	341.97	110	171.42	55	327.55	107
10^4	142.82	79	*	*	831.67	219	293.24	76	820.57	219
rand1										
2×10^3	66.92	24	*	*	760.06	388	161.81	76	726.43	378
4×10^3	162.20	29	*	*	1786.76	496	258.74	69	1695.18	479
6×10^3	252.81	30	*	*	2010.50	368	316.25	54	1931.67	364
8×10^3	386.67	35	*	*	3063.53	435	544.61	72	2771.14	405
10^4	†	†	*	*	4012.57	451	694.83	70	3747.25	429
rand2										
2×10^3	164.18	22	*	*	5821.79	889	427.70	67	5378.99	844
4×10^3	†	†	*	*	3881.73	397	451.03	43	4086.40	424
6×10^3	†	†	†	†	7555.65	522	901.72	51	6908.09	485
8×10^3	†	†	†	†	8678.47	478	883.14	41	8171.98	456
10^4	†	†	†	†	15984.27	698	1536.32	60	15664.92	696

Table 9: Solution times and iteration counts for generated problems

* Failed to solve due to timeout

† Failed to solve due to insufficient memory

Scen	IPM Time	IPM Iter	Simplex Time	Simplex Iter	Benders Time	Benders Iter	Level Time	Level Iter	Lin. Damp. Time	Lin. Damp. Iter
saphir										
50	‡	‡	255.03	73918	465.18	113	396.59	76	474.50	121
100	‡	‡	916.04	143194	701.14	120	533.06	60	671.41	119
200	‡	‡	7579.14	385231	‡	‡	2555.47	206	‡	‡
500	†	†	†	†	2556.06	115	2339.76	59	2570.49	115
1000	†	†	†	†	4294.47	109	4650.19	78	4472.34	115

Table 10: Solution times and iteration counts for SAMPL problems

* Failed to solve due to timeout

† Failed to solve due to insufficient memory

‡ Failed to solve due to numerical difficulties

Scen	IPM Time	IPM Iter	Simplex Time	Simplex Iter	Benders Time	Benders Iter	Level Time	Level Iter	Lin. Damp. Time	Lin. Damp. Iter
WATSON.I										
128	1.71	33	1.44	7985	0.92	1	0.92	1	0.92	1
256	3.89	38	3.66	15818	1.58	1	1.59	1	1.58	1
512	8.91	47	5.88	30237	2.62	1	2.61	1	2.63	1
1024	20.27	54	14.44	60941	5.12	1	5.31	1	5.15	1

Table 11: Solution times and iteration counts for two-stage approximations of WATSON problems

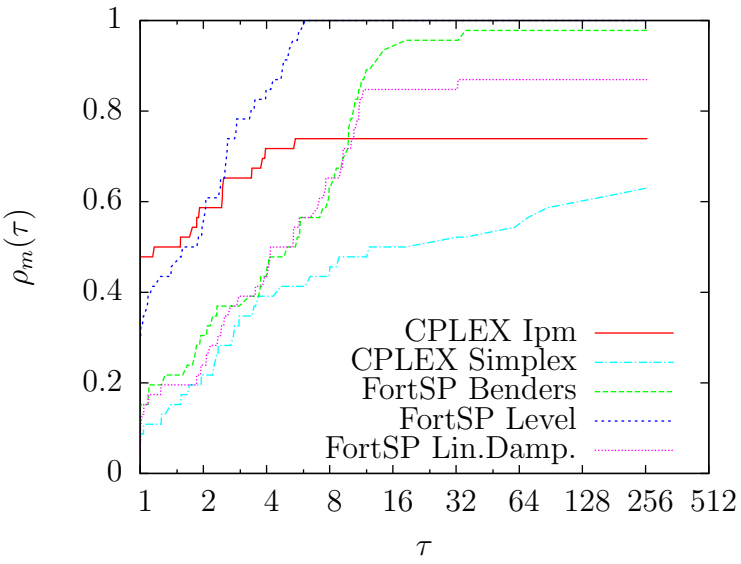

Figure 3: Performance profile in a $\log_2$ scale

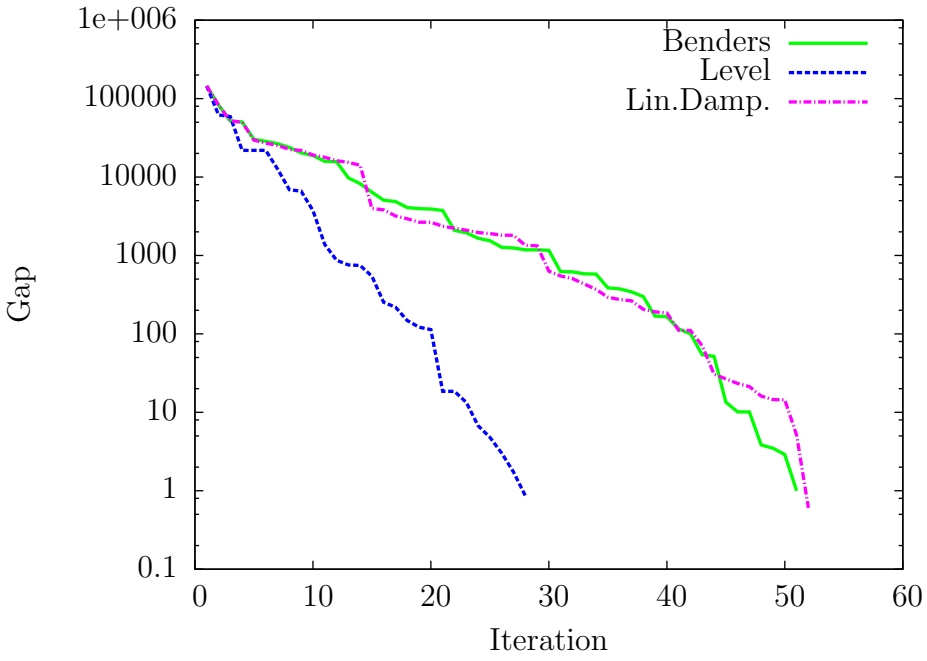

Figure 4: Gap between lower and upper bounds for stormG2-1000 problem

i.e. the method terminated if $(z^* - z_*)/(|z_*| + 0.1) \leq \varepsilon$, where z_* and z^* are, respectively, lower and upper bounds on the value of the objective function. For CPLEX barrier optimizer the default complementarity tolerance was used as a stopping criterion.

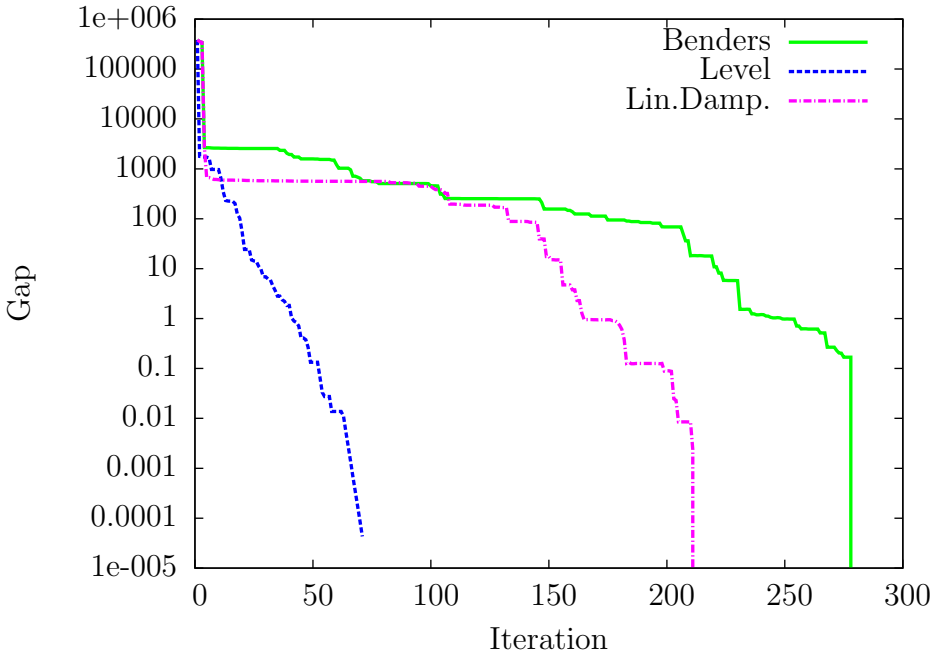

Figure 5: Gap between lower and upper bounds for 4node-32768 problem

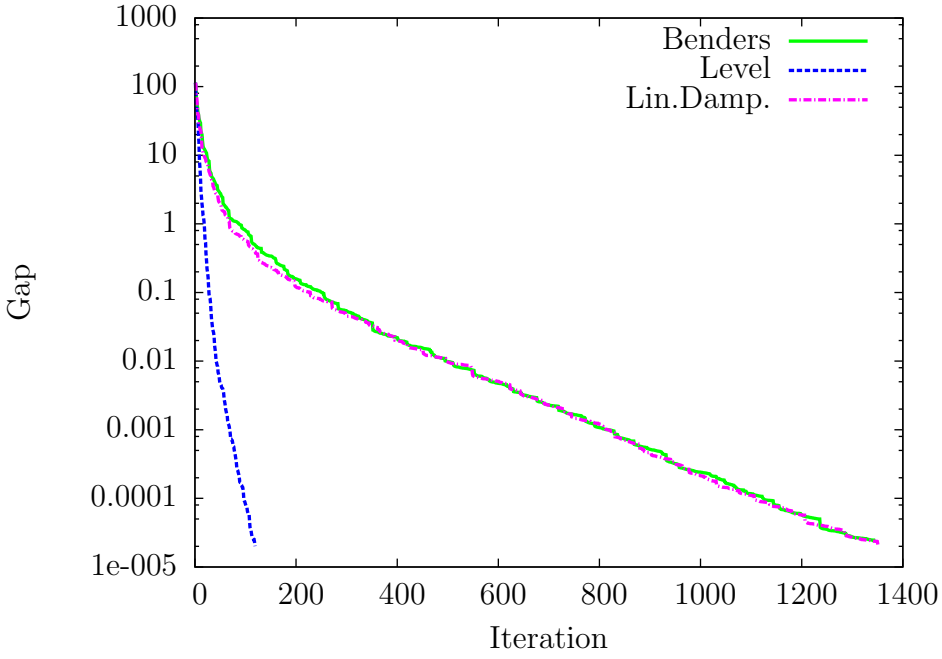

Figure 6: Gap between lower and upper bounds for rand2-10000 problem

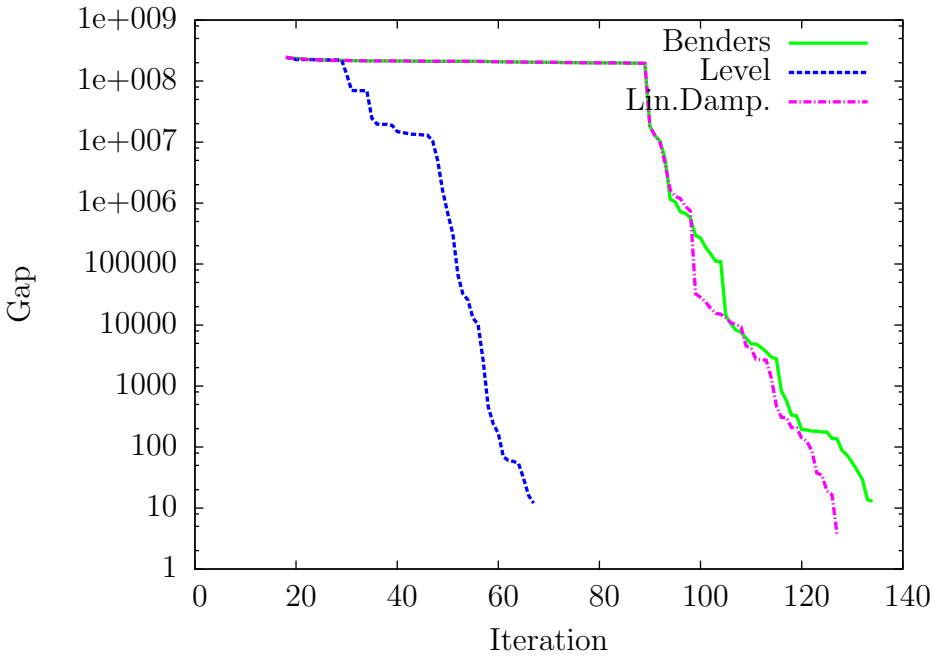

Figure 7: Gap between lower and upper bounds for saphir-1000 problem

5 Discussion and conclusion

In this paper we make a case for continuing research and development of solution algorithms for processing scenario based SP recourse problems in particular two stage SPs. Our empirical computational study clearly establishes the need for robust solution methods which can process diverse SP applications in particular as these scale up in size and number of scenarios. We show that simple use of even most powerful hypersparse solvers cannot process many industrial strength models specially, when the model sizes scale up due to multiple scenarios. We also observe that the interior point method outperformed simplex in the majority of cases. In our experiments Benders decomposition performs well, however, through the regularisation by level decomposition we are able to process very large instances of SP application models. We hope to report similar study for two stage integer stochastic programming benchmark models.

References

Ariyawansa, K. A. and Felt, A. J. (2004). On a new collection of stochastic linear programming test problems. *INFORMS Journal on Computing*, 16(3):291–299.

Beale, E. M. L. (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society, Series B*, 17:173–184.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252. Re-publised in *Computational Management Science 2* (2005), 3–19.

Birge, J. R., Dempster, M. A. H., Gassmann, H. I., Gunn, E. A., King, A. J., and Wallace, S. W. (1987). A standard input format for multiperiod stochastic linear programs. *COAL Newsletter*, 17:1–19.

Birge, J. R. and Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392.

Birge, J. R. and Louveaux, F. V. (1997). *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Colombo, M., Gondzio, J., and Grothey, A. (2006). A warm-start approach for large-scale stochastic linear programs. Technical report, School of Mathematics The University of Edinburgh.

Consigli, G. and Dempster, M. A. H. (1998). Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81:131–162.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1:197–206.

Dantzig, G. B. and Madansky, A. (1961). On the solution of two-stage linear programs under uncertainty. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 165–176. University of California Press, Berkeley.

Dantzig, G. B. and Wolfe, P. (1960). The decomposition principle for linear programs. *Operations Research*, 8:101–111.

Di Domenica, N., Lucas, C., Mitra, G., and Valente, P. (2009). Stochastic programming and scenario generation within a simulation framework: An information perspective. *IMA Journal of Management Mathematics*, 20:1–38.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Ellison, E. F. D., Hajian, M., Jones, H., Levkovitz, R., Maros, I., Mitra, G., and Sayers, D. (2008). *FortMP Manual*. Brunel University: London, Numerical Algorithms Group: Oxford. `http://www.optirisk-systems.com/manuals/FortmpManual.pdf`.

Fábián, C. I. (2000). Bundle-type methods for inexact data. *Central European Journal of Operations Research*, 8:35–55. Special issue, T. Csendes and T. Rapcsák, eds.

Fábián, C. I. and Szőke, Z. (2007). Solving two-stage stochastic programming problems with level decomposition. *Computational Management Science*, 4:313–353.

Fabozzi, F. J., Focardi, S., and Jonas, C. (2007). Trends in quantitative equity management: survey results. *Quantitative Finance*, 7(2):115–122.

Gassmann, H. I. and Wallace, S. W. (1996). Solving linear programs with multiple right-hand sides: Pricing and ordering schemes. *Annals of Operations Research*, 64:237–259.

Holmes, D. (1995). A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS). `http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html`.

Kall, P. and Mayer, J. (1998). On testing SLP codes with SLP-IOR. In Giannessi, F., Rapcsák, T., and Komlósi, S., editors, *New Trends in Mathematical Programming: Homage to Steven Vajda*, pages 115–135. Kluwer Academic Publishers.

Kall, P. and Mayer, J. (2005). *Stochastic Linear Programming: Models, Theory, and Computation*. Springer, International Series in Operations Research and Management Science.

Kall, P. and Wallace, S. W. (1994). *Stochastic Programming*. John Wiley & Sons, Chichester.

Kiwiel, K. C. (1985). *Methods of descent for nondifferentiable optimization*. Springer-Verlag, Berlin, New York.

König, D., Suhl, L., and Koberstein, A. (2007). Optimierung des Gasbezugs im liberalisierten Gasmarkt unter Berücksichtigung von Röhren- und Untertagespeichern. In *Sammelband zur VDI Tagung "Optimierung in der Energiewirtschaft" in Leverkusen*.

Lemaréchal, C. (1978). Nonsmooth optimization and descent methods. *Research Report* 78-4, IIASA, Laxenburg, Austria.

Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69:111–147.

Lougee-Heimer, R. (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47(1):57–66.

Mayer, J. (1998). *Stochastic Linear Programming Algorithms*. Gordon and Breach Science Publishers, Amsterdam.

Mitra, G., Di Domenica, N., Birbilis, G., and Valente, P. (2007). Stochastic programming and scenario generation within a simulation framework: An information perspective. *Decision Support Systems*, 42:2197–2218.

Prékopa, A. (1995). *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht.

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:877–898.

Ruszczyński, A. (1986). A regularized decomposition method for minimizing the sum of polyhedral functions. *Mathematical Programming*, 35:309–333.

Ruszczyński, A. (2003). Decomposition methods. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming, Handbooks in Operations Research and Management Science*, volume 10, pages 141–211. Elsevier, Amsterdam.

Ruszczyński, A. and Shapiro, A. (2003). Stochastic programming models. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming, Handbooks in Operations Research and Management Science*, volume 10, pages 1–64. Elsevier, Amsterdam.

Ruszczyński, A. and Świętanowski, A. (1997). Accelerating the regularized decomposition method for two-stage stochastic linear problems. *European Journal of Operational Research*, 101:328–342.

Valente, C., Mitra, G., Sadki, M., and Fourer, R. (2009). Extending algebraic modelling languages for stochastic programming. *Informs Journal on Computing*, 21(1):107–122.

Valente, P., Mitra, G., Poojari, C., Ellison, E. F., Di Domenica, N., Mendi, M., and Valente, C. (2008). *SAMPL/SPInE User Manual*. OptiRisk Systems. `http://www.optirisk-systems.com/manuals/SpineAmplManual.pdf`.

Van Slyke, R. and Wets, R. J. B. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663.

Wallace, S. W. and Ziemba, W. T., editors (2005). *Applications of Stochastic Programming*. Society for Industrial and Applied Mathematic.

Wets, R. J. B. (1974). Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review*, 16:309–339.